Rapport du Projet GSE5 2019/2020

Introduction

L'objectif du projet est de concevoir un système de cartographie en temps réel. Le dispositif d'acquisition est basé sur un scanner laser (Lidar) contrôlé par une carte Exynos qui peut à la fois communiquer via une connexion série / USB. Les données Lidar sont transmises à une station de base PC exécutant les logiciels SLAM et de visualisation. Le périphérique d'acquisition et la station de base communiquent à l'aide d'un système d'exploitation de robot (ROS) et d'une connexion Wifi. L'algorithme SLAM est basé sur Google Cartographer et sur un outil de visualisation 2D/3D appelé RVIZ, tous les deux intégrés dans ROS. Nous voulons aussi une fonctionnalité qui nous permet de détecter des objets par un webcam intégré lorsque le système est un cours de tracer la carte. Enfin, nous espérons réaliser une telle fonction de navigation sur la carte qui a été réalisé.

High-Level design

Hardware

Ce projet est lié principalement sur l'appareil « Lidar », il est une technique de mesure à distance fondée sur l'analyse des propriétés d'un faisceau de lumière renvoyé vers son émetteur. Cette technique est largement utilisée dans différents domaines, par exemple topographie et sciences de l'environnement, plus particulièrement dans le guidage automatique pour les voitures.

Pour atteindre l'objectif, les appareils sont placés sur différentes couches, chaque couche est supportée par une maille métallique, les couches sont soutenues par des vis. La surface des mailles métalliques est recouverte par un film plastique pour éviter les courts-circuits. Afin de permettre au lidar de mieux obtenir les données scannées, nous choisissons de placer le lidar dans la couche la plus haute. La carte STM32 est placée dans la couche juste au-dessous du lidar, cette carte va servir à une autre partie du projet (IMU). La webcam et le module wifi sont placé à la même couche que Exyons, sachant que ces deux appareils utilisent des ports du Exynos. Et à la fin est la couche des batteries.

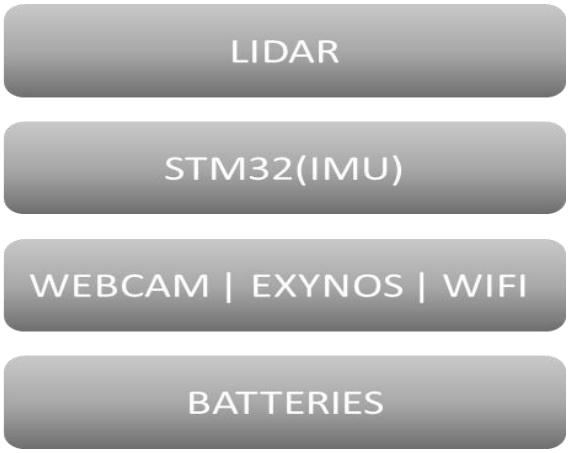

Figure 1 la répartition des couches

Pour réaliser ce système au niveau hardware, chaque groupe a sa propre tâche, Dans notre groupe, nous avons implémenté le lidar, et fixé trois batteries en parallèle par des vis. Les écrous originaux du lidar n'adaptent pas les vis que nous utilisent, et donc nous avons déplacé les écrous originaux du lidar par les nouveaux écrous.

	Modèle	**Société de production**
LIDAR	Rplidar A1	SLAMTEC
STM32	STM32F411E-DISCO	STMicroelectronics
WEBCAM	Dualpix	HERCULES
EXYNOS	Odroid-XU4	Hardkernel
WIFI	Odroid WiFi Module 3	Hardkernel
BATTERIE	JY-5567100	liquidware

Ce tableau ci-dessus nous donne ensemble des modèles de tous les appareils que nous avons utilisé.

Software

La partie de logiciel de ce projet est basée sur Linux : ubuntu mate 16.04 32 bits sur la carte odroid et ubuntu18.04 sur notre PC.

Le logiciel global utilisé dans ce projet est ROS (Robot operating system). ROS est un système pour développer le robot qui fournit des librairies et des outils, y compris les pilotes des périphériques, librairies des fonctions et outil du visualisation. Quatre versions de ROS sont disponibles maintenant : indigo, kinetic, lunar, melodic. La version de melodic (supporte ubuntu 18.04) est installée sur notre PC et la version de kinetic(supporte ubuntu 16.04) est installée sur notre carte odroid.

Rplidar

Le package « rplidar_ros » est utilisé pour piloter la fonction de RPLIDAR sous ROS. Ce package fournit une gestion de base des périphériques pour le scanner laser 2D RPLIDAR et il peut envoyer les données comme un rostopic de type ***/scan***.

SLAM cartographer

SLAM(Simultaneouslocalization and mapping) consiste, pour un robot ou véhicule autonome, à simultanément construire ou améliorer une carte de son environnement et de s'y localiser. Hector_mapping, gmapping et cartographer sont les trois méthodes les plus utilisées. Cartographer de Google est utilisé dans notre projet.

Par rapport aux deux autres algorithmes, le cartographe n'a pas besoin de s'appuyer sur un odomètre. En même temps, grâce à l'utilisation de la détection de boucle, son erreur est plus faible, et il n'a pas besoin d'un radar haute fréquence pour obtenir de meilleurs résultats.

Le package « cartographer_ros » permet d'intégrer l'algorithme cartographer à l'environnement de ROS. Selon les erreurs rencontrées lors de l'installation, le « proto3 » doit être installé correctement avant de construire ce package.

Webcam

Le package usb_cam, qui peut recevoir les données de caméra par port USB et les envoyer comme rostopic de type ***image_raw***, est utilisé pour piloter V4L USB Camera sous ROS. Afin de réduire la quantité de données et accélérer le transfert, nous transférons les images en utilisant un format compressé.

Communication par WiFi

Un avantage de ROS est que les données des périphériques sont transmises par nœud (topic). Il permet d'utiliser les données dans différentes machines par la connexion au réseau. SSH connexion, une fonction de Linux, avec la configuration maître et esclave sous ROS sont utilisés pour réaliser cette communication. La carte odroid est définie comme maître et notre PC est défini comme esclave. Notre carte et notre PC doivent être dans le même réseau en utilisant SSH. Une fois la connexion est établie, nous pouvons voir tous les « topic » publié par maître dans notre PC quand nous utilisons la commande « rostopic ».

Opencv

Dans cette partie, ce que nous voulons réaliser est la détection des objets sur vidéo en temps réel.

OpenCV est une librairie pour vision par ordinateur, l'architecture DARKNET et modèle YOLO marchent basé sur cette librairie. L'algorithme YOLO (You Only Look Once) est un modèle de réseau de détection d'objets. Par rapport à d'autres algorithmes, Yolo ne calcule l'image entière qu'une seule fois, la quantité de calcul est réduite et ses performances en temps réel sont plus élevées. Il existe une façon d'utiliser CUDA pour accélérer la vitesse de calcul, mais il ne supporte que les GPUs de Nvidia.

Le package *darknet_ros* permet d'intégrer Darknet, un framework de réseau neuronal, dans l'environnement de ROS. Le message d'image peut être transmis entre ROS et OpenCV par CvBridge. Le rostopic ***image_raw*** envoyé par webcam est transmis à OpenCV et renvoyé à ROS en ajoutant les résultats de la détection.

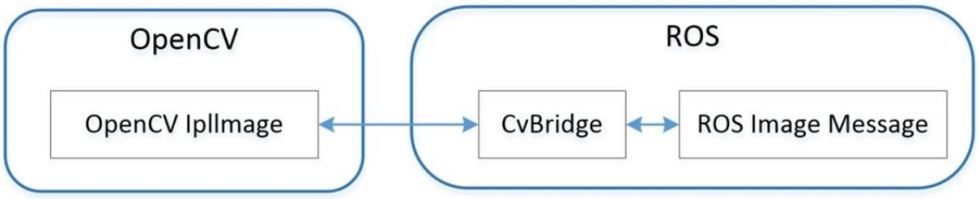

Figure 2 communication entre ROS et OpenCV

Le darknet_ros ne supporte pas encore le YOLOv3 maintenant, le YOLOv2-tiny est utilisé pour identifier le type d'objet détecté. Par rapport au YOLOv2, la précision de détection est réduite, mais la vitesse est améliorée. À la fin, il peut atteindre à une vitesse de détection autour d'un frame par seconde.

Navigation

Turtlebot_gazebo est un package pour la simulation de navigation sous ROS. Nous pouvons créer un scénario dans Gazebo et créer la carte et faire la navigation selon ce scénario en utilisant le package. Aussi, la navigation peut être créé par un fichier existant de la carte avec AMCL, un système de localisation probabiliste pour un robot se déplaçant en 2D. Dans cette partie, nous avons réalisé la navigation selon un scénario. Malheureusement, le fichier de la carte de bâtiment E n'a pas créé finalement à cause de WiFi instable, nous n'avons pas réussi de faire cette navigation.

Script

Lors de l'appel de différents appareils, beaucoup de terminales doivent être ouverts. Et il faut configurer l'environnement et remarcher le « ssh » chaque fois. Pour simplifier les étapes de taper les commandes, nous avons écrit certains scripts. La commande « mate-terminal -t » est utilisée pour ouvrir un nouvel terminal dan ubuntu mate et « gnome-terminal -t » dans ubuntu. Et le mot de passe est requis pour marcher « ssh », un fichier de type expect pour réaliser une connexion automatique de « ssh » en utilisant script. En utilisant ces scripts, nous avons seulement besoin de taper la commande «. /run » dans la carte et notre PC.

Test et résultat

Après que chaque couche d'équipement est assemblée, nous avons fait les tests de chaque étape du projet. Au début, notre système était alimenté par une batterie, mais nous avons constaté que le système entrerait dans un état de redémarrage juste après le démarrage, car une batterie ne peut pas fournir assez de courant, et nous avons envisagé de mettre trois batteries en parallèle.

Test du RpLidar

Lorsque le Lidar est activé sans utiliser le SLAM, il n'affiche que les points (point rouge) détectée par Lidar.

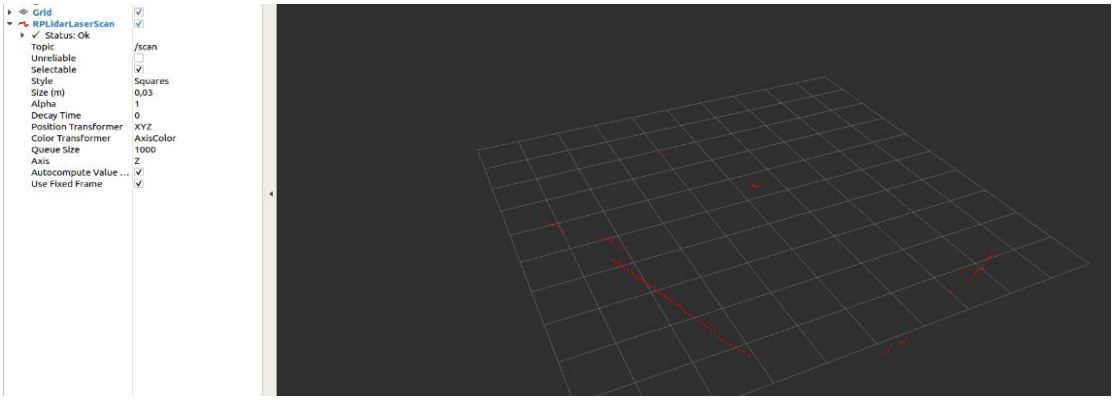

Figure 3 Test du RpLidar

Test du cartographer

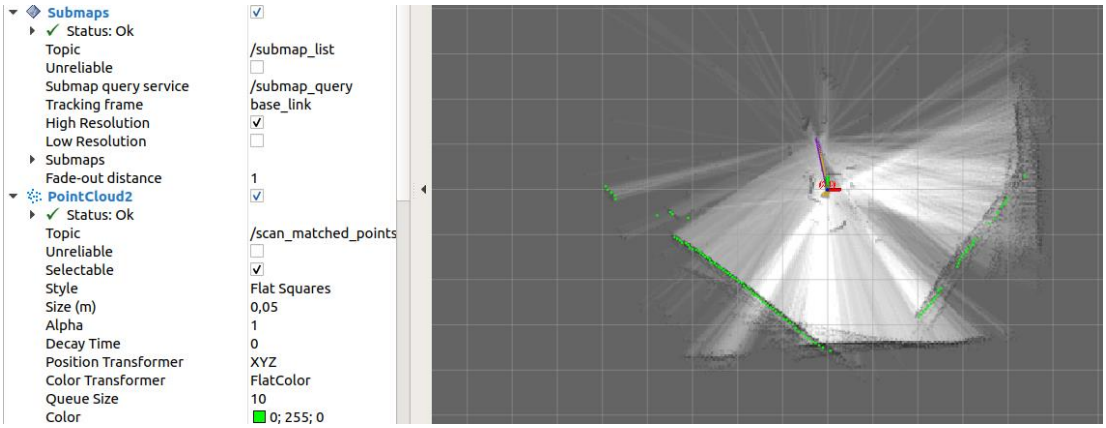

Figure 4 Test du cartographer

Test global (détection des objets intégré dans RVIZ)

Dans le test de cette partie, nous avons ajouté la détection des objets en temps réel et intégré l'affichage dans RVIZ.

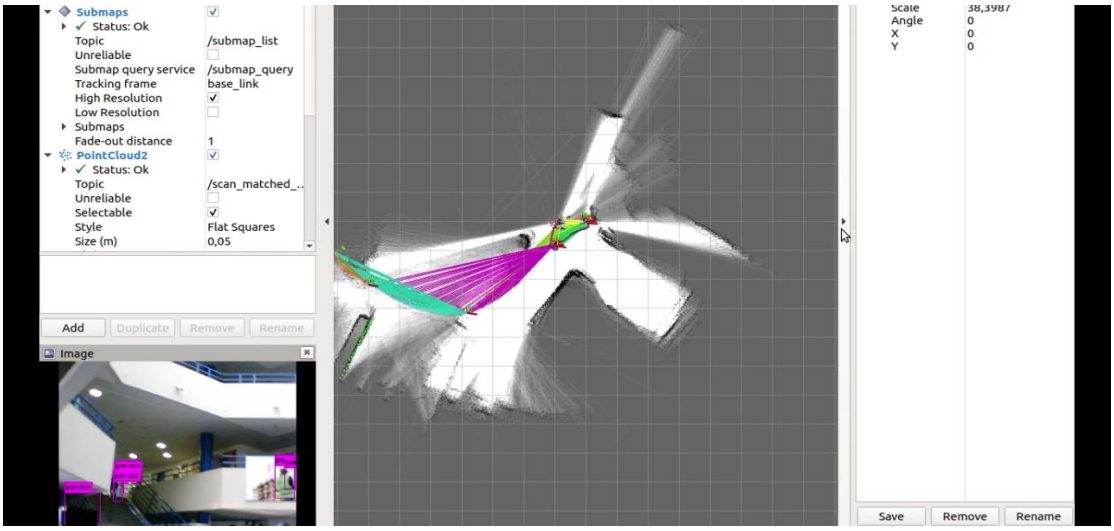

Figure 5 Test global

La carte du bâtiment E

Après tous les tests, nous avons fait une démonstration qui trace la carte de l'étage -2, bâtiment E.

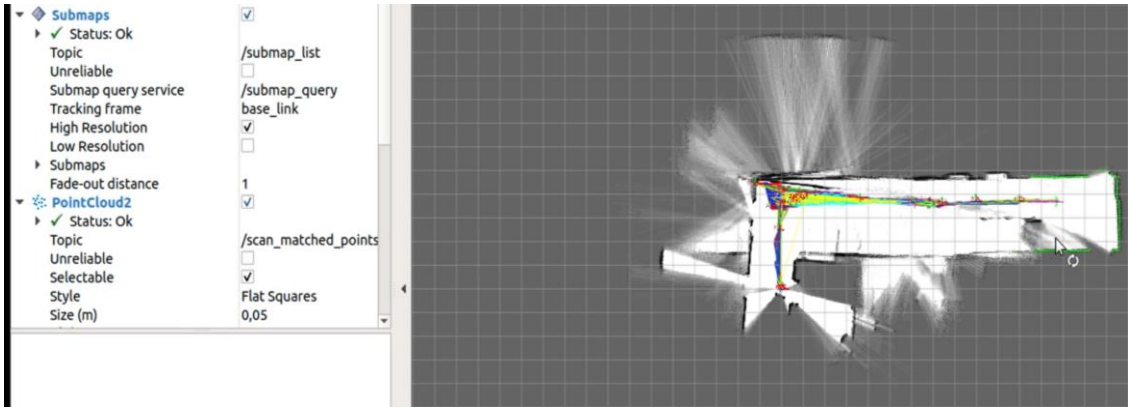

Figure 6 La carte du bâtiment E

Conclusion

À la fin, nous avons réalisé de faire la cartographie du rez-de-chaussée de bâtiment O, en considérant le problème de connexion au réseau, nous avons tracé aussi une carte pour le bâtiment sud étage 0, en même temps, notre système de robot peut renvoyer la vidéo au PC via wifi, du côté PC, il traite ce flux de vidéo en détectant les objets. Une fois la carte est faite, nous pouvons donc enregistrer des données cartographiques, y compris des donnés vidéo.

Grâce à ce projet, nous avons mis en place un système de cartographie basé sur « Lidar » et « Exynos », qui comprend les niveaux matériels et logiciels. Nous avons déterminé la cartographie des étages de l'école, nous avons réussi partiellement la partie « Navigation » en raison d'une mauvaise gestion du temps du projet.

Compte tenu de l'imperfection du système, il a encore besoin de nombreuses améliorations en termes de performances et de fonctionnalités. Nous avons énuméré les points suivants.

1. Nous espérons ajouter quatre roues au-dessous de ce système afin que le robot puisse décider de la direction de sa propre progression à travers la navigation.
2. Deuxièmement, étant donné que le réseau de l'école n'est pas très stable, et la transmission de données vidéo et de données lidar en même temps via le WiFi n'est pas une très bonne solution. Donc il faut trouver une solution pour résoudre ce problème.

Reference

Mate : https://wiki.odroid.com/getting_started/os_installation_guide#tab__odroid-xu4

ROS : https://www.ros.org

Rplidar : http://wiki.ros.org/rplidar

OpenCV : https://opencv.org

http://www.guyuehome.com/2988

Yolo : http://wiki.ros.org/darknet_ros

SLAM : https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

Cartographer : https://google-cartographer-ros.readthedocs.io/en/latest/index.html

Webcam: http://wiki.ros.org/usb_cam

Navigation: http://wiki.ros.org/amcl

http://wiki.ros.org/turtlebot_simulator/Tutorials/hydro/Make%20a%20map%20and%20navigate%20with%20it